
SPECpy Documentation

Release 1

Lakshmipriya Sukumar and Brian Toby

September 20, 2012

CONTENTS

1	<i>Module spec: SPEC-like emulation</i>	1
2	<i>Global variables</i>	3
3	<i>All Functions</i>	5
	Index	11

CHAPTER
ONE

MODULE SPEC: SPEC-LIKE EMULATION

Python functions listed below are designed to emulate similar commands/macros in SPEC.

Motor interface routines.

Description	Relative	Absolute
move motor	mvr ()	mv ()
move motor with wait	umvr ()	umv ()
where is this motor?		wm ()
where are all motors?		wa ()

Scaler routines

description	command
start and readout scaler after completion	ct ()
start scaler and return	count_em ()
wait for scaler to complete	wait_count ()
read scaler	get_counts ()

GLOBAL VARIABLES

COUNT defines the default counting time (sec) when ct is called without an argument. Defaults to 1 sec.

MAX_RETRIES Number of times to retry an EPICS operation (that are nominally expected to work on the first try) before generating an exception.

DEBUG Set to True for code development/testing use only. Causes lots of print statements to be executed.

ENABLE Initialized as False, which indicates that EPICS PV access should be simulated (also allows module to be imported for documentation generation, etc. without importing PyEpics). Use function EnableEPICS() to set ENABLE to True.

ALL FUNCTIONS

The functions available in this module are listed below.

`spec.DefineMtr(symbol, prefix, comment='')`

Define a motor for use in this module. Adds a motor to the motor table.

Parameters

- **symbol** (*string*) – a symbolic name for the motor. A global variable is defined in this module’s name space with this name. This must be unique; exception specException is raised if a name is reused.
- **prefix** (*string*) – the prefix for the motor PV (ioc:mnnn). Omit the motor record field name (.VAL, etc.).
- **comment** (*string*) – a human-readable text field that describes the motor. Suggestion: include units and define the motion direction.

Returns value of entry created in motor table.

If you will use the “`from <module> import *`” python command to import these routines into the current module’s name space, it is necessary to repeat this command after `DefineScaler()` to import the globals defined within in the top namespace:

Example (recommended for interactive use):

```
>>> from spec import *
>>> EnableEPICS()
>>> DefineMtr('mtrXX1','ioc1:mtr98','Example motor #1')
>>> DefineMtr('mtrXX2','ioc1:mtr99','Example motor #2')
>>> from spec import *
>>> mv(mtrXX1, 0.123)
```

Note that if the second `from ... import *` command is not used, the variables `mtrXX1` and `mtrXX2` cannot be accessed and the final command will fail.

Alternate example (preferred for use in code):

```
>>> import spec
>>> spec.EnableEPICS()
>>> spec.DefineSpec('mtrXX1','ioc1:mtr98','Example motor #1')
>>> spec.DefineMtr('mtrXX2','ioc1:mtr99','Example motor #2')
>>> spec.mv(spec.mtrXX1, 0.123)
```

`spec.DefineScaler(prefix, channels=8, index=0)`

Defines a scaler to be used for this module

Parameters

- **prefix** (*string*) – the prefix for the scaler PV (ioc:mnnn). Omit the scaler record field name (.CNT, etc.)
- **channels** (*int*) – the number of channels associated with the scaler. Defaults to 8.
- **index** (*int*) – an index for the scaler, if more than one will be defined. The default (0) is used to define the scaler that will be used when `ct()` is called with one or no arguments.

Example (recommended for interactive use):

```
>>> from spec import *
>>> EnableEPICS()
>>> DefineScaler('id1:scaler1', 16)
>>> DefineScaler('id1:scaler2', index=1)
>>> ct()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

Alternate example (preferred for use in code):

```
>>> import spec as ct
>>> ct.EnableEPICS()
>>> ct.DefineScaler('ioc1:3820:scaler1', 16)
>>> ct.DefineScaler('ioc1:3820:scaler2', index=1)
>>> ct.ct()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
>>> ct.ct(index=1)
[1, 2, 3, 4, 5, 6, 7, 8]
```

`spec.EnableEPICS(state=True)`

Call to enable communication with EPICS.

If not called then the module will function in simulation mode only. If the PyEpics module cannot be loaded, then simulation will also be used.

Parameters `state` (*bool*) – if False is specified, then simulation mode is used (default value, True)

`spec.ExplainMtr(mtr)`

Show the description for a motor, as defined in `DefineMtr()`

Parameters `mtr` (*various*) – symbolic name for the motor, can take two forms.

Type	Description
str	interpreted as a global symbol
int	value references an entry in mtrDB

Returns motor description (str) or ‘?’ if not defined

`spec.ListMtrs()`

Returns a list of the variables defined as motor symbols.

Returns a python list of defined motor symbols (list of str values).

`spec.PositionMtr(mtr, pos, wait=True)`

Move a motor

Position a motor associated with mtr to position pos, wait for the move to complete if wait is True, or else return immediately. The function attempts to verify the move command has been acted upon.

Parameters

- **mtr** (*int*) – a value corresponding to an entry in the motor table, as defined in `DefineMtr()`. If the value does not correspond to a motor entry, an exception is raised.

- **pos** (*float*) – a value to position the motor. If the value is invalid or outside the limits an exception occurs (todo: are hard limits checked?).
- **wait** (*bool*) – a flag that specifies if the move should be completed before the function returns. If False, the function returns immediately.

`spec.ReadMtr (mtr)`

Return the motor position associated with the passed motor value.

Parameters **mtr** (*int*) – a value corresponding to an entry in the motor table. If the value does not correspond to a motor entry, an exception is raised.

Returns motor position (*float*).

`spec.count_em (count=None, index=0)`

Cause scaler to start counting for specified period.

Counting must be on time. Counting on a monitor is not implemented.

count values for the channels (see `DefineScaler ()`)

Parameters

- **count-time** (*float*) – time (sec) to count, if omitted COUNT is used
- **index** (*int*) – an index for the scaler, if more than one will be defined (see `DefineScaler ()`). The default (0) is used if not specified.

Returns None

Example:

`>>> count_em()`

`spec.ct (count=None, index=0)`

Cause scaler to count for specified period.

Counting must be on time. Counting on a monitor is not implemented.

count values for the channels (see `DefineScaler ()`)

Parameters

- **count-time** (*float*) – time (sec) to count, if omitted COUNT is used
- **index** (*int*) – an index for the scaler, if more than one will be defined (see `DefineScaler ()`). The default (0) is used if not specified.

Returns a list of channels values

Example:

`>>> ct()`
[1, 2, 3, 4, 5, 6, 7, 8]

`spec.get_counts (wait=False)`

Read scaler with optional delay, must follow `count_em`

reads count values for the channels (see `DefineScaler ()`)

Parameters **wait** (*bool*) – True causes the routine to wait for the scaler to complete; False (default) will read the scaler instantaneously

Returns a list of channels values

Example:

```
>>> get_counts()
[1, 2, 3, 4, 5, 6, 7, 8]
```

`spec.mv(mtr, pos)`

Move motor without wait

If the move cannot be made, an exception is raised.

Parameters

- **mtr** (*int*) – a value corresponding to an entry in the motor table, as defined in `DefineMtr()`. If the value does not correspond to a motor entry, an exception is raised.
- **pos** (*float*) – a value to position the motor. If the value is invalid or outside the limits, an exception occurs.

Example:

```
>>> mv(samX, 0.1)
```

`spec.mvr(mtr, delta)`

Move motor relative to current position without wait.

If the move cannot be made, an exception is raised.

Parameters

- **mtr** (*int*) – a value corresponding to an entry in the motor table, as defined in `DefineMtr()`. If the value does not correspond to a motor entry, an exception is raised.
- **delta** (*float*) – a value to offset the motor. If the resulting value is invalid or outside the limits, an exception occurs.

Example:

```
>>> mvr(samX, 0.1)
```

`spec.umv(mtr, pos)`

Move motor with wait.

If the move cannot be completed, an exception is raised.

Parameters

- **mtr** (*int*) – a value corresponding to an entry in the motor table, as defined in `DefineMtr()`. If the value does not correspond to a motor entry, an exception is raised.
- **pos** (*float*) – a value to position the motor. If the value is invalid or outside the limits, an exception occurs.

Example:

```
>>> umv(samX, 0.1)
```

`spec.umvr(mtr, delta)`

Move motor relative to current position with wait.

If the move cannot be completed, an exception is raised.

Parameters

- **mtr** (*int*) – a value corresponding to an entry in the motor table, as defined in `DefineMtr()`. If the value does not correspond to a motor entry, an exception is raised.
- **delta** (*float*) – a value to offset the motor. If the resulting value is invalid or outside the limits, an exception occurs.

Example:

```
>>> umvr(samX, 0.1)
```

`spec.wa` (*label=False*)

Print positions of all motors defined using `DefineMtr()`.

Parameters **label** (*bool*) – a flag that specifies if the list should include the motor descriptions. If omitted or False, the descriptions are not included.

Example:

```
>>> wa()
samX          1.0
samZ          0.0
>>> wa(True)
samX          1.0      sample X position (mm) + outboard
samZ          0.0      sample Z position (mm) + up
```

`spec.wait_count()`

Wait for scaler to finish, must follow `count_em`

Returns None

Example:

```
>>> wait_count()
```

`spec.wm (*mtrs)`

Read out specified motor(s).

Arguments one or more motor table entries that are defined in `DefineMtr()`.

Returns a single float if a single argument is passed to `wm`. Returns a list of floats if more than one argument is passed.

Example:

```
>>> wm(samX, samZ)
[1.0, 0.0]
```


INDEX

C

count_em() (in module spec), [7](#)
ct() (in module spec), [7](#)

D

DefineMtr() (in module spec), [5](#)
DefineScaler() (in module spec), [5](#)

E

EnableEPICS() (in module spec), [6](#)
ExplainMtr() (in module spec), [6](#)

G

get_counts() (in module spec), [7](#)

L

ListMtrs() (in module spec), [6](#)

M

mv() (in module spec), [8](#)
mvr() (in module spec), [8](#)

P

PositionMtr() (in module spec), [6](#)

R

ReadMtr() (in module spec), [7](#)

U

umv() (in module spec), [8](#)
umvr() (in module spec), [8](#)

W

wa() (in module spec), [9](#)
wait_count() (in module spec), [9](#)
wm() (in module spec), [9](#)